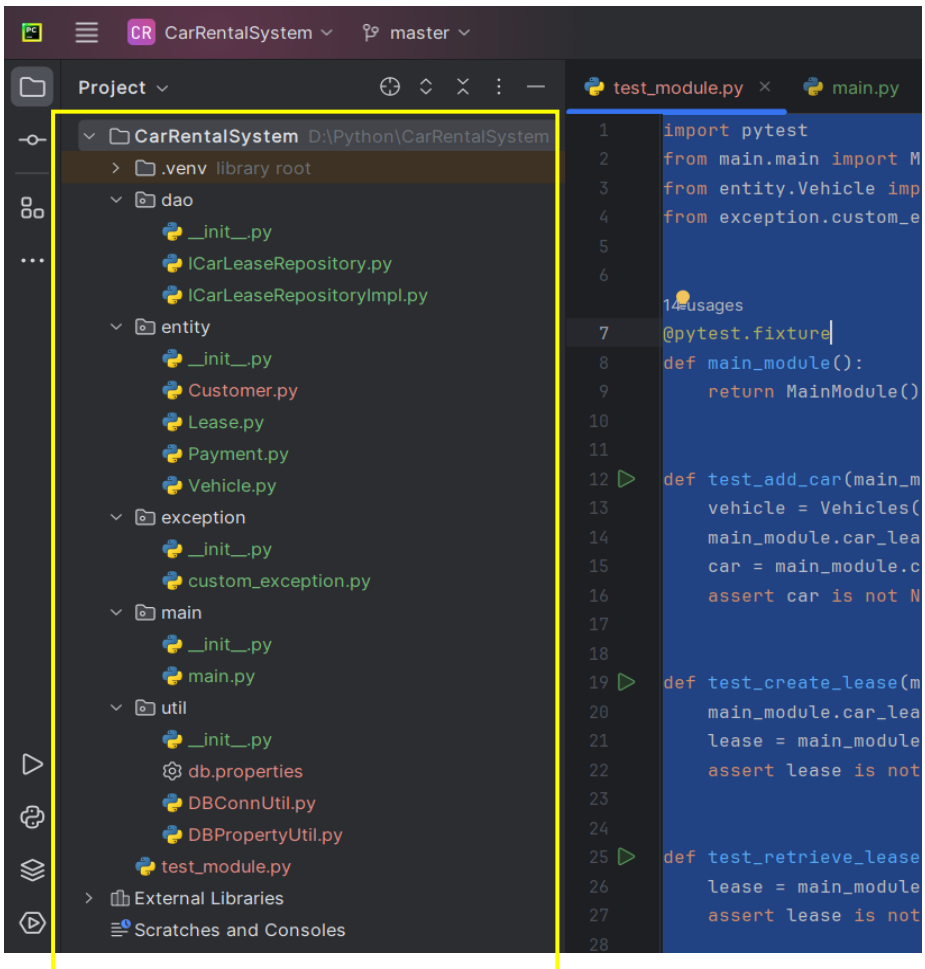
NAME : - BISWARUP ROY

CASE STUDY ON

CAR RENTAL SYSTEM

CAR RENTAL SYSTEM

- The following **Directory structure** is to be followed in the application.
  - **entity/model**
    - Create entity classes in this package. All entity class should not have any business logic.
  - **dao**
    - Create Service Provider interface to showcase functionalities.
    - Create the implementation class for the above interface with db interaction.
  - **exception**
    - Create user defined exceptions in this package and handle exceptions whenever needed.
  - **util**
    - Create a **DBPropertyUtil** class with a static function which takes property file name as parameter and returns connection string.
    - Create a **DBConnUtil** class which holds **static method** which takes connection string as parameter file and returns **connection object(Use method defined in DBPropertyUtil class to get the connection String)**.
  - **main**
    - Create a class MainModule and demonstrate the functionalities in a menu driven application.



(Structure)

Create following tables in SQL Schema with appropriate class and write the unit test case for the Car Rental application.

```
mysql> CREATE DATABASE carRentalSystem;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| careerhub          |
| carrentalsystem    |
| college            |
| hexprac            |
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| school             |
| sisdb              |
| sql_hr             |
| sql_inventory      |
| sql_invoicing      |
| sql_store          |
| sys                |
| techshop           |
| ticketbookingsystem|
| world              |
+--------------------+
18 rows in set (0.01 sec)
```

Schema Design:

1. **Vehicle Table:**
   - vehicleID (Primary Key)
   - make
   - model
   - year
   - dailyRate
   - status (available, notAvailable)
   - passengerCapacity
   - engineCapacity

```
mysql> USE carrentalsystem;
Database changed
mysql> CREATE TABLE vehicle (
    ->     vehicleID INT PRIMARY KEY AUTO_INCREMENT,
    ->     make VARCHAR(50),
    ->     model VARCHAR(50),
    ->     year INT(4),
    ->     dailyRate DECIMAL(10,2),
    ->     status ENUM('available', 'not available'),
    ->     passengerCapacity INT,
    ->     engineCapacity INT);
Query OK, 0 rows affected, 1 warning (0.05 sec)
```

2. **Customer Table:**
    - customerID (Primary Key)
    - firstName
    - lastName
    - email
    - phoneNumber

```
mysql> CREATE TABLE Customer(customerID INT PRIMARY KEY AUTO_INCREMENT,
    -> firstName VARCHAR(50),
    -> lastName VARCHAR(50),
    -> email VARCHAR(100),
    -> phoneNumber INT(10));
Query OK, 0 rows affected, 1 warning (0.01 sec)
```

3. **Lease Table:**
    - leaseID (Primary Key)
    - vehicleID (Foreign Key referencing Vehicle Table)
    - customerID (Foreign Key referencing Customer Table)
    - startDate
    - endDate
    - type (to distinguish between DailyLease and MonthlyLease)

```
mysql> CREATE TABLE Lease(leaseID INT PRIMARY KEY AUTO_INCREMENT,
    -> vehicleID INT,
    -> customerID INT,
    -> startDate DATE,
    -> endDate DATE,
    -> type ENUM('DailyLease','MonthlyLease'),
    -> FOREIGN KEY (vehicleID) REFERENCES Vehicle(vehicleID),
    -> FOREIGN KEY (customerID) references Customer(customerID));
Query OK, 0 rows affected (0.04 sec)
```

4. **Payment Table:**
    - paymentID (Primary Key)
    - leaseID (Foreign Key referencing Lease Table)
    - paymentDate
    - amount

```
mysql> CREATE TABLE Payment(paymentID INT PRIMARY KEY AUTO_INCREMENT,
    -> leaseID INT,
    -> paymentDate DATE,
    -> amount DECIMAL(10,2),
    -> FOREIGN KEY (leaseID) REFERENCES Lease (leaseID));
Query OK, 0 rows affected (0.03 sec)
```

5. **Create the model/entity classes corresponding to the schema within package entity with variables declared private, constructors(default and parametrized) and getters,setters)**

test_module.py | Customer.py | Lease.py | Payment.py | Vehicle.py | main.py

```python
4 usages  new *
class Vehicles:
    new *
    def __init__(self, make, model, year, daily_rate, status, passenger_capacity, engine_capacity):
        self.__make = make
        self.__model = model
        self.__year = year
        self.__dailyRate = daily_rate
        self.__status = status
        self.__passengerCapacity = passenger_capacity
        self.__engineCapacity = engine_capacity

2 usages (1 dynamic)  new *
@property
def make(self):
    return self.__make

1 usage (1 dynamic)  new *
@make.setter
def make(self, val):
    self.__make = val

2 usages (1 dynamic)  new *
@property
def model(self):
    return self.__model

1 usage (1 dynamic)  new *
@model.setter
def model(self, new_model):
    self.__model = new_model

2 usages (1 dynamic)  new *
@property
def year(self):
    return self.__year

2 usages (1 dynamic)  new *
@property
def dailyRate(self):
    return self.__dailyRate

1 usage (1 dynamic)  new *
@dailyRate.setter
def dailyRate(self, new_rate):
    self.__dailyRate = new_rate

2 usages (1 dynamic)  new *
@property
def status(self):
    return self.__status

1 usage (1 dynamic)  new *
@status.setter
def status(self, new_status):
    self.__status = new_status

2 usages (1 dynamic)  new *
@property
def passengerCapacity(self):
    return self.__passengerCapacity

1 usage (1 dynamic)  new *
@passengerCapacity.setter
def passengerCapacity(self, passenger_capacity):
    self.__passengerCapacity = passenger_capacity

2 usages (1 dynamic)  new *
@property
def engineCapacity(self):
    return self.__engineCapacity

1 usage (1 dynamic)  new *
@engineCapacity.setter
def engineCapacity(self, engine_capacity):
    self.__engineCapacity = engine_capacity
```

(Constructor)

```
class Customers:
    def __init__(self, first_name, last_name, email, phone_number):
        self.__firstName = first_name
        self.__lastName = last_name
        self.__email = email
        self.__phoneNumber = phone_number
```

(Getter & Setters)

```
    3 usages (2 dynamic)
    @property
    def firstName(self):
        return self.__firstName

    2 usages (2 dynamic)
    @firstName.setter
    def firstName(self, first_name):
        self.__firstName = first_name

    3 usages (2 dynamic)
    @property
    def lastName(self):
        return self.__lastName

    2 usages (2 dynamic)
    @lastName.setter
    def lastName(self, last_name):
        self.__lastName = last_name
```

```
    3 usages (2 dynamic)
    @property
    def email(self):
        return self.__email

    2 usages (2 dynamic)
    @email.setter
    def email(self, set_email):
        self.__email = set_email

    3 usages (2 dynamic)
    @property
    def phoneNumber(self):
        return self.__phoneNumber

    2 usages (2 dynamic)
    @phoneNumber.setter
    def phoneNumber(self, phome_number):
        self.__phoneNumber = phome_number
```

(Constructor)

test_module.py | Lease.py | Payment.py | main.py

```
new *
class Lease:
    new *
    def __init__(self, vehicle_id, customer_id, start_date, end_date, vehicle_type):
        self.__vehicleID = vehicle_id
        self.__customerID = customer_id
        self.__startDate = start_date
        self.__endDate = end_date
        self.__type = vehicle_type

```

(Getters & Setter)

```
1 usage  new *
@property
def vehicleID(self):
    return self.__vehicleID

new *
@vehicleID.setter
def vehicleID(self, vehicle_id):
    self.__vehicleID = vehicle_id

1 usage  new *
@property
def customerID(self):
    return self.__customerID

new *
@customerID.setter
def customerID(self, customer_id):
    self.__customerID = customer_id

1 usage  new *
@property
def startDate(self):
    return self.__startDate
```

```
new *
@startDate.setter
def startDate(self, start_date):
    self.__startDate = start_date

1 usage  new *
@property
def endDate(self):
    return self.__endDate

new *
@endDate.setter
def endDate(self, end_date):
    self.__endDate = end_date

1 usage  new *
@property
def type(self):
    return self.__type

new *
@type.setter
def type(self, new_type):
    self.__type = new_type
```

(Constructor)

```
new *
class Payments:
    new *
    def __init__(self, payment_id, lease_id, payment_date, amount):
        self.__paymentID = payment_id
        self.__leaseID = lease_id
        self.__paymentDate = payment_date
        self.__amount = amount
```

(Getters & Setters)

```
new *
@property
def paymentID(self):
    return self.__paymentID

1 usage  new *
@property
def leaseID(self):
    return self.__leaseID

new *
@leaseID.setter
def leaseID(self, lease_id):
    self.__leaseID = lease_id

1 usage  new *
@property
def paymentDate(self):
    return self.__paymentDate
```

```
new *
@paymentDate.setter
def paymentDate(self, payment_date):
    self.__paymentDate = payment_date

1 usage  new *
@property
def amount(self):
    return self.__amount

new *
@amount.setter
def amount(self, new_amount):
    self.__amount = new_amount
```

6. **Service Provider Interface/Abstract class:**
   Keep the interfaces and implementation classes in package dao
   - Create Interface for **ICarLeaseRepository** and add following methods which interact with database.

7. Implement the above interface in a class called **ICarLeaseRepositoryImpl in package dao.**

- **Car Management**
  1. **addCar(Car car)**
     parameter : Car
     return type : void
  2. **removeCar()**
     parameter : carID

(ICarleaseRepository)

```python
from abc import ABC, abstractmethod
from typing import List
from entity import Customer, Lease, Vehicle


2 usages  new *
class ICarLeaseRepository(ABC):
    1 usage (1 dynamic)  new *
    @abstractmethod
    def addCar(self, vehicle: Vehicle) -> None:
        pass
```

(ICarleaseRepositoryImpl)

```python
    2 usages (1 dynamic)  new *
def addCar(self, vehicle: Vehicle):
    cursor = self.connection.cursor()
    query = ("INSERT INTO Vehicle (make, model, year, dailyRate, status, passengerCapacity, engineCapacity) "
             "VALUES (%s, %s, %s, %s, %s, %s, %s)")
    data = (vehicle.make, vehicle.model, vehicle.year, vehicle.dailyRate, vehicle.status, vehicle.passengerCapacity,
            vehicle.engineCapacity)
    cursor.execute(query, data)
    self.connection.commit()
    cursor.close()


    1 usage  new *
def removeCar(self, vehicle_id: int):
    cursor = self.connection.cursor()
    query = "DELETE FROM Vehicle WHERE vehicleID = %s"
    cursor.execute(query, (vehicle_id,))
    self.connection.commit()
    cursor.close()
```

3. listAvailableCars() -
   parameter: NIL
   return type: return List of Car
4. listRentedCars() – return List of Car
   parameter: NIL
   return type: return List of Car
5. findCarById(int carID) – return Car if found or throw exception
   parameter: NIL
   return type: return List of Car

(ICarleaseRepository)

```
        new *
        @abstractmethod
        def removeCar(self, vechile_id: int) -> None:
            pass

        new *
        @abstractmethod
        def listAvailableCars(self) -> List[Vehicle]:
            pass

        new *
        @abstractmethod
        def listRentedCars(self) -> List[Vehicle]:
            pass

        2 usages (2 dynamic)  new *
        @abstractmethod
        def findCarById(self, vechile_id: int) -> Vehicle:
            pass
```

(ICarleaseRepositoryImpl)

```
    def listAvailableCars(self) -> List[Vehicle]:
        cursor = self.connection.cursor()
        query = "SELECT * FROM Vehicle WHERE status = 'available'"
        cursor.execute(query)
        vehicles = cursor.fetchall()
        cursor.close()
        return vehicles

    1 usage  new *
    def listRentedCars(self) -> List[Vehicle]:
        cursor = self.connection.cursor()
        query = "SELECT * FROM Vehicle WHERE status = 'notAvailable'"
        cursor.execute(query)
        rented_cars = cursor.fetchall()
        cursor.close()
        return rented_cars

    3 usages (2 dynamic)  new *
    def findCarById(self, vehicle_id: int) -> Vehicle:
        cursor = self.connection.cursor()
        query = "SELECT * FROM Vehicle WHERE vehicleID = %s"
        cursor.execute(query, (vehicle_id,))
        vehicle = cursor.fetchone()
        cursor.close()
        if vehicle:
            return vehicle
        else:
            raise CarNotFoundException("Car not found with ID: {}".format(vehicle_id))
```

- **Customer Management**
    1. addCustomer(Customer customer)
        parameter : Customer
        return type : void
    2. void removeCustomer(int customerID)
        parameter : CustomerID
        return type : void
    3. listCustomers()
        parameter : NIL
        return type : list of customer
    4. findCustomerById(int customerID)
        parameter : CustomerID
        return type : Customer

(ICarleaseRepositoryImpl)

```
1 usage  new *
def addCustomer(self, customer: Customer) -> None:
    cursor = self.connection.cursor()
    query = ("INSERT INTO Customer (firstName, lastName, email, phoneNumber) "
             "VALUES (%s, %s, %s, %s)")
    data = (customer.firstName, customer.lastName, customer.email, customer.phoneNumber)
    cursor.execute(query, data)
    self.connection.commit()
    cursor.close()

1 usage  new *
def removeCustomer(self, customer_id: int) -> None:
    cursor = self.connection.cursor()
    query = "DELETE FROM Customer WHERE customerID = %s"
    cursor.execute(query, (customer_id,))
    self.connection.commit()
    cursor.close()

1 usage  new *
def listCustomers(self) -> List[Customer]:
    cursor = self.connection.cursor()
    query = "SELECT * FROM Customer"
    cursor.execute(query)
    customers = cursor.fetchall()
    cursor.close()
    return customers

2 usages (1 dynamic)  new *
def findCustomerById(self, customer_id: int) -> Customer:
    cursor = self.connection.cursor()
    query = "SELECT * FROM Customer WHERE customerID = %s"
    cursor.execute(query, (customer_id,))
    customer_data = cursor.fetchone()
```

- **Lease Management**
    1. createLease()
        parameter : int customerID, int carID, Date startDate, Date endDate
        return type : Lease
    2. void returnCar();
        parameter : int leaseID
        return type : Lease info
    3. List<Lease> listActiveLeases();
        parameter : NIL
        return type : Lease list
    4. listLeaseHistory();
        parameter : NIL
        return type : Lease list

(ICarleaseRepositoryImpl)

```
    def createLease(self, vehicle_id: int, customer_id: int, start_date: str, end_date: str, v_type):
        cursor = self.connection.cursor()
        query = ("INSERT INTO Lease (vehicleID, customerID, startDate, endDate, type) "
                 "VALUES (%s, %s, %s, %s, %s)")
        data = (vehicle_id, customer_id, start_date, end_date, v_type)
        cursor.execute(query, data)
        self.connection.commit()
        cursor.close()

    1 usage  new *
    def returnCar(self, lease_id: int):
        cursor = self.connection.cursor()
        query = "UPDATE Lease SET endDate = CURDATE() WHERE leaseID = %s"
        cursor.execute(query, (lease_id,))
        self.connection.commit()
        cursor.close()

    1 usage  new *
    def listActiveLeases(self) -> List[Lease]:
        cursor = self.connection.cursor()
        query = "SELECT * FROM Lease WHERE CURDATE() >= startDate AND CURDATE() <= endDate"
        cursor.execute(query)
        active_leases = cursor.fetchall()
        cursor.close()
        return active_leases

    1 usage  new *
    def listLeaseHistory(self) -> List[Lease]:
        cursor = self.connection.cursor()
        query = "SELECT * FROM Lease"
        cursor.execute(query)
        lease_history = cursor.fetchall()
        cursor.close()
```

- **Payment Handling**
    1. void recordPayment();
       parameter : Lease lease, double amount
       return type : void

(ICarleaseRepository)

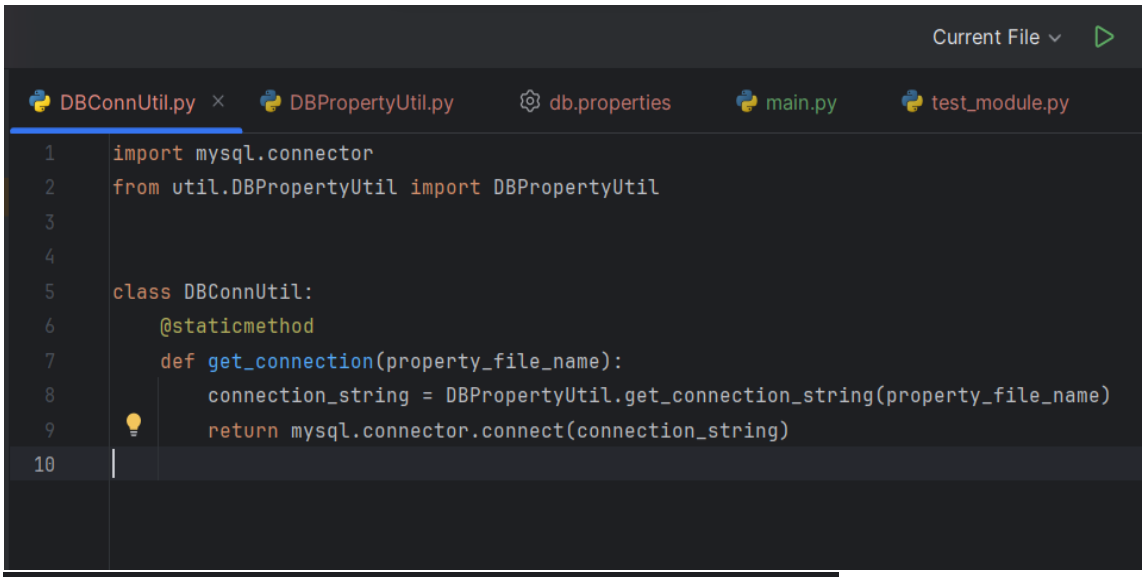
```
58

    new *
59  @abstractmethod
60  def recordPayment(self, lease: Lease, date: str, amount: float) -> None:
61      pass
62
```

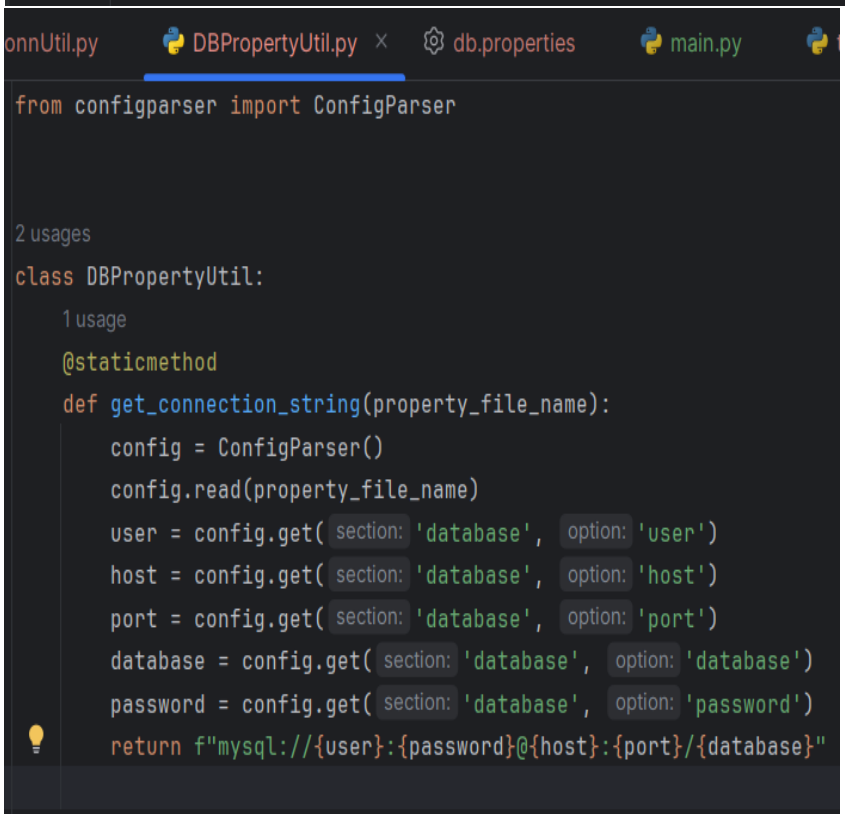
(ICarleaseRepositoryImpl)

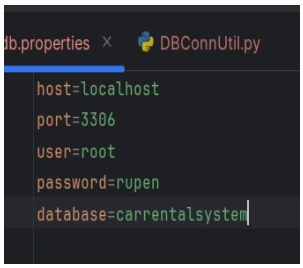
```
138

    1 usage  new *
139 def recordPayment(self, lease: int, date: str, amount: float) -> None:
140     cursor = self.connection.cursor()
141     query = ("INSERT INTO Payment (leaseID, paymentDate, amount) "
142              "VALUES (%s, %s, %s)")
143     data = (lease, date, amount)
144     cursor.execute(query, data)
145     self.connection.commit()
146     cursor.close()
147
```

Connect your application to the SQL database:

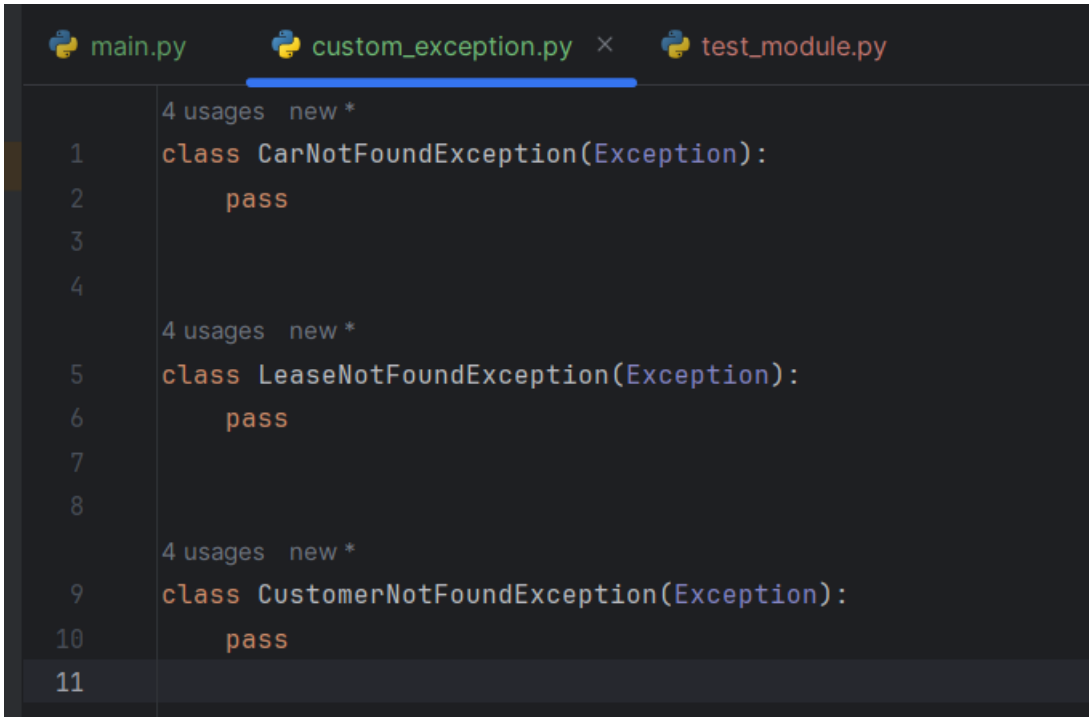
8. Connect your application to the SQL database and write code to establish a connection to your SQL database.
    - Create a utility class **DBConnection** in a package **util** with a static variable **connection** of Type **Connection** and a static method **getConnection()** which returns connection.
    - Connection properties supplied in the connection string should be read from a property file.
    - Create a utility class **PropertyUtil** which contains a static method named **getPropertyString()** which reads a property fie containing connection details like hostname, dbname, username, password, port number and returns a connection string.







9. Create the exceptions in package **myexceptions** and create the following custom exceptions and throw them in methods whenever needed. Handle all the exceptions in main method,
    - **CarNotFoundException**: throw this exception when user enters an invalid car id which doesn't exist in db.
    - **LeaseNotFoundException**: throw this exception when user enters an invalid lease id which doesn't exist in db.
    - **CustomerrNotFoundException**: throw this exception when user enters an invalid customer id which doesn't exist in db.



Unit Testing:

10. Create Unit test cases for **Ecommerce System** are essential to ensure the correctness and reliability of your system. Following questions to guide the creation of Unit test cases:
    - Write test case to test car created successfully or not.
    - Write test case to test lease is created successfully or not.
    - Write test case to test lease is retrieved successfully or not.
    - write test case to test exception is thrown correctly or not when customer id or car id or lease id not found in database.

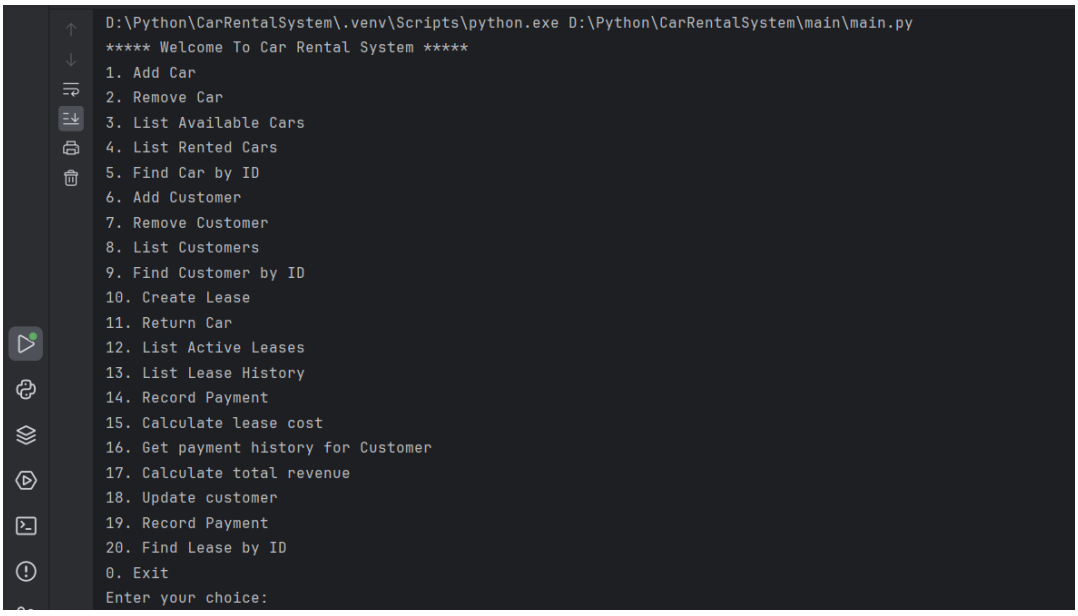
main.py | test_module.py | ICarLeaseRepositoryImpl.py | ICarLeaseRepository.py

```python
from unittest.mock import Mock, MagicMock

import pytest

from entity.Lease import Lease
from main.main import MainModule
from entity.Vehicle import Vehicles
from exception.custom_exception import CarNotFoundException, CustomerNotFoundException, LeaseNotFoundException


16 usages
@pytest.fixture
def main_module():
    return MainModule()


def test_add_car(main_module):
    mock_cursor = Mock()
    mock_connection = Mock()
    mock_connection.cursor.return_value = mock_cursor
    main_module.car_lease.connection = mock_connection
    vehicle = Vehicles( make: "Toyota", model: "Corolla", year: 2022, daily_rate: 175.00, status: "available",
                        passenger_capacity: 5, engine_capacity: 1.8)
    main_module.car_lease.addCar(vehicle)
    mock_cursor.execute.assert_called_once()
    mock_connection.commit.assert_not_called()

    car = main_module.car_lease.findCarById(1)
    assert car is not None
```

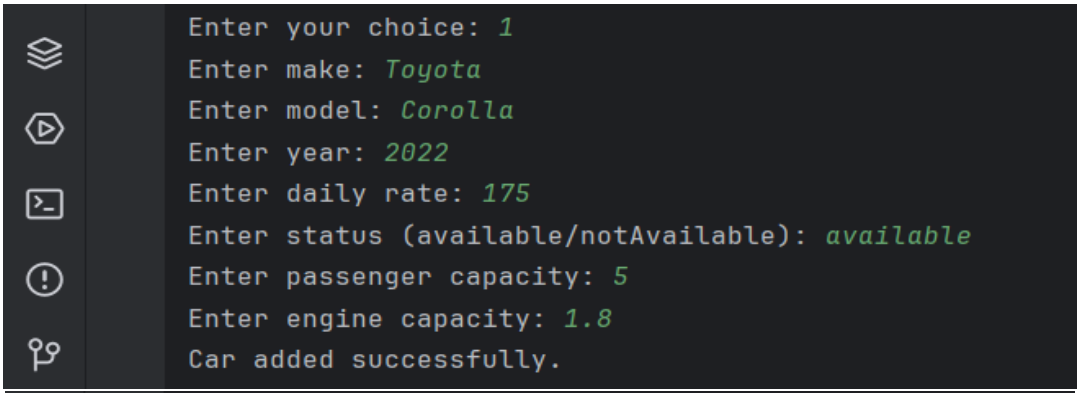
```python
def test_create_lease(main_module):
    mock_cursor = Mock()
    mock_connection = Mock()
    mock_connection.cursor.return_value = mock_cursor
    main_module.car_lease.connection = mock_connection
    lease = Lease( vehicle_id: 1, customer_id: 1, start_date: "2024-02-05", end_date: "2024-02-06", vehicle_type: "DailyLease")
    main_module.car_lease.createLease(lease)
    mock_cursor.execute.assert_called_once()
    mock_connection.commit.assert_not_called()

    lease_found = main_module.car_lease.findLeaseById(1)
    assert lease_found is not None


def test_retrieve_lease(main_module):
    lease = main_module.car_lease.findLeaseById(1)
    assert lease is not None


def test_car_not_found_exception(main_module):
    with pytest.raises(CarNotFoundException):
        main_module.car_lease.findCarById(1000)


def test_customer_not_found_exception(main_module):
    with pytest.raises(CustomerNotFoundException):
        main_module.car_lease.findCustomerById(1000)


def test_lease_not_found_exception(main_module):
    with pytest.raises(LeaseNotFoundException):
        main_module.car_lease.findLeaseById(1000)
```

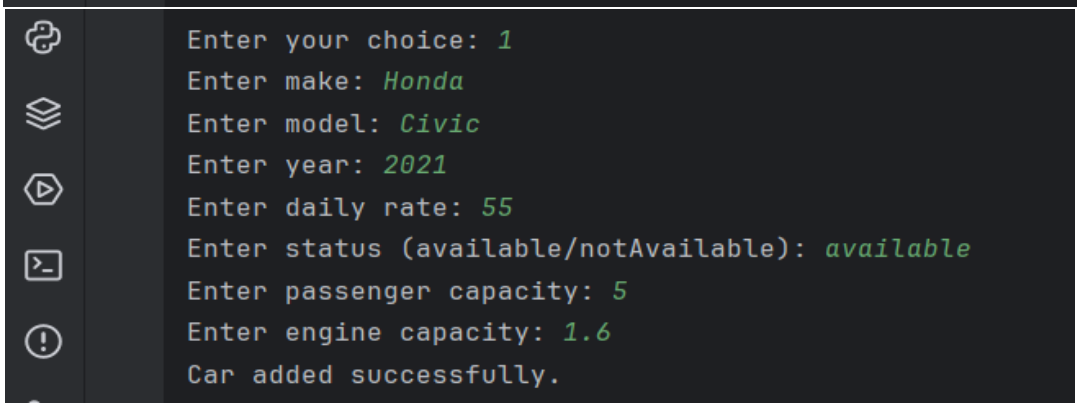
Let’s Run Our Car Rental System

- **Dashboard**



- **Let’s add 5 cars in our database**





```
Enter your choice: 1
Enter make: Ford
Enter model: Fusion
Enter year: 2023
Enter daily rate: 85
Enter status (available/notAvailable): not available
Enter passenger capacity: 5
Enter engine capacity: 2
Car added successfully.
```

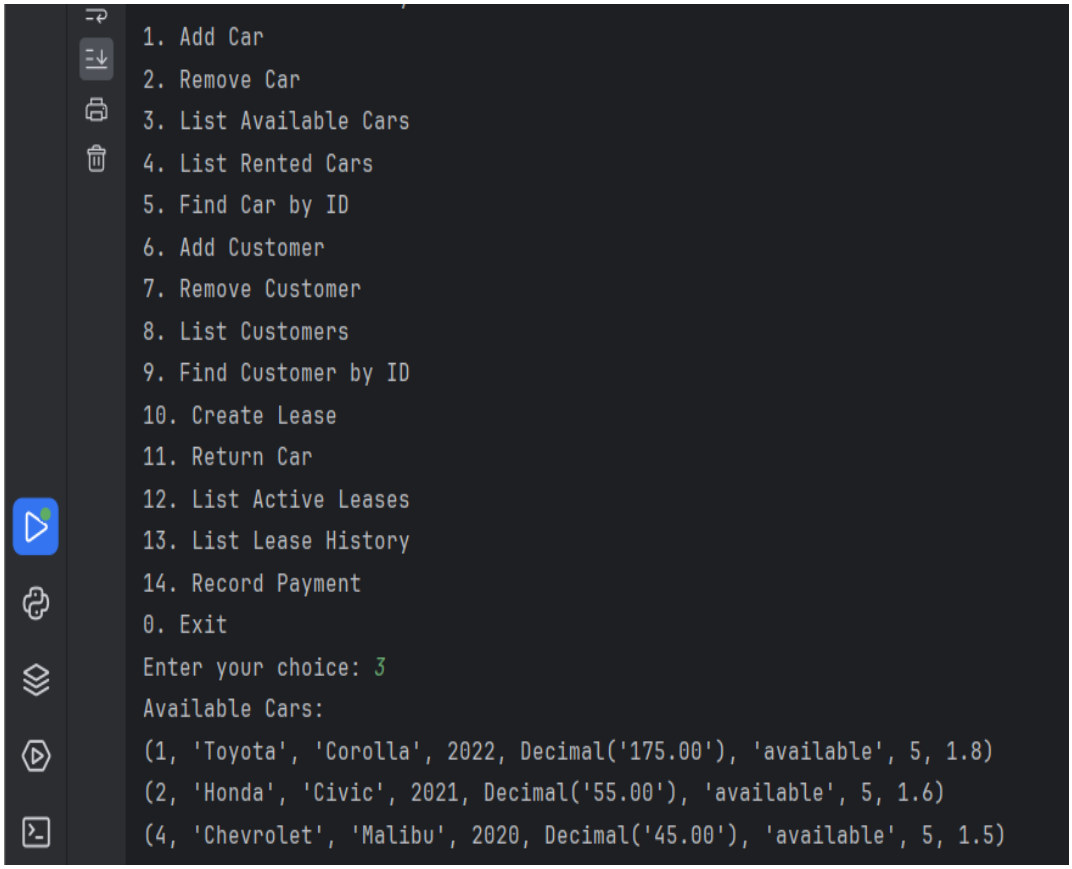
```
Enter your choice: 1
Enter make: Chevrolet
Enter model: Malibu
Enter year: 2020
Enter daily rate: 45
Enter status (available/notAvailable): available
Enter passenger capacity: 5
Enter engine capacity: 1.5
Car added successfully.
```

```
Enter your choice: 1
Enter make: Nissan
Enter model: Altima
Enter year: 2022
Enter daily rate: 58
Enter status (available/notAvailable): not available
Enter passenger capacity: 4
Enter engine capacity: 2
Car added successfully.
```

- **Let's check in MySQL Workbench**

```
mysql> SELECT * FROM Vehicle;
+-----------+----------+---------+------+----------+--------------+-----------------+----------------+
| vehicleID | make     | model   | year | dailyRate | status       | passengerCapacity | engineCapacity |
+-----------+----------+---------+------+----------+--------------+-----------------+----------------+
|         1 | Toyota   | Corolla | 2022 |    175.00 | available    |               5 |            1.8 |
|         2 | Honda    | Civic   | 2021 |     55.00 | available    |               5 |            1.6 |
|         3 | Ford     | Fusion  | 2023 |     85.00 | not available |               5 |              2 |
|         4 | Chevrolet | Malibu | 2020 |     45.00 | available    |               5 |            1.5 |
|         5 | Nissan   | Altima  | 2022 |     58.00 | not available |               4 |              2 |
+-----------+----------+---------+------+----------+--------------+-----------------+----------------+
5 rows in set (0.00 sec)
```

- **List all available car**



- **Let's check in MySQL Workbench**

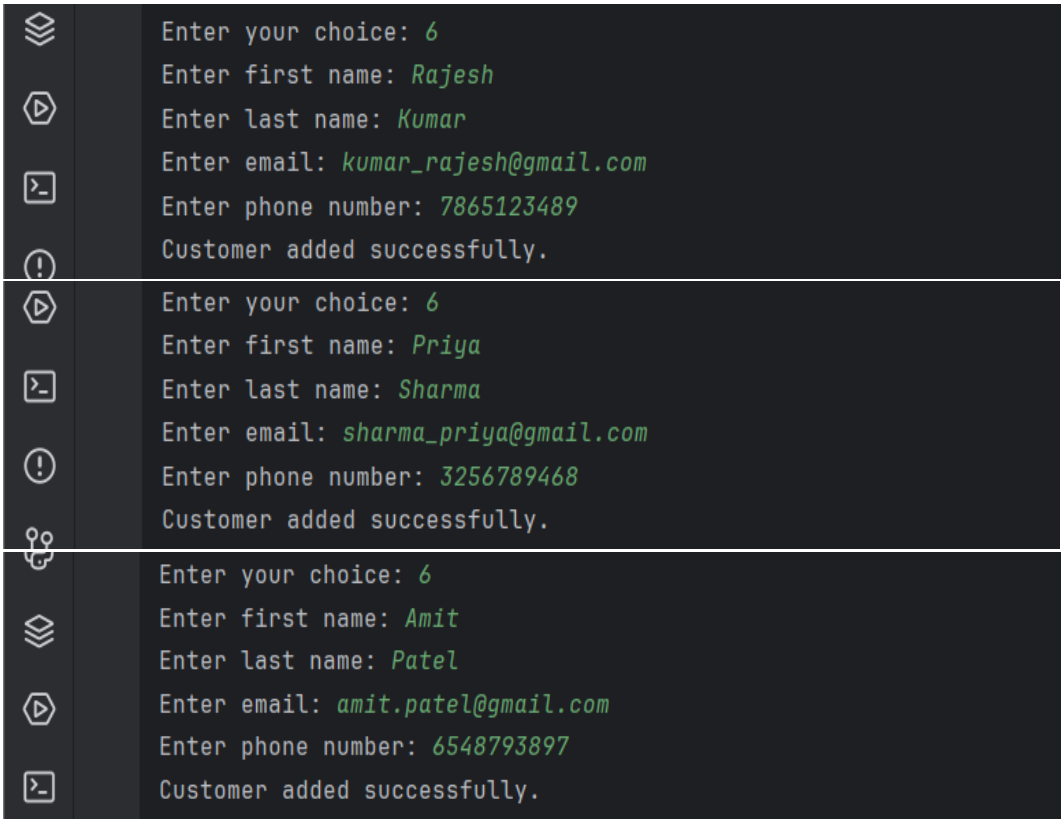
```
mysql> SELECT vehicleID , make , model , status
    -> from vehicle
    -> where status = 'available';
+-----------+-----------+---------+-----------+
| vehicleID | make      | model   | status    |
+-----------+-----------+---------+-----------+
|         1 | Toyota    | Corolla | available |
|         2 | Honda     | Civic   | available |
|         4 | Chevrolet | Malibu  | available |
+-----------+-----------+---------+-----------+
3 rows in set (0.00 sec)
```

- **Let's Find a car by its ID**

- **Let's check in MySQL Workbench**

```
mysql> SELECT * FROM vehicle
    -> WHERE vehicleID = 3;
+-----------+------+--------+------+-----------+---------------+-----------------+----------------+
| vehicleID | make | model  | year | dailyRate | status        | passengerCapacity | engineCapacity |
+-----------+------+--------+------+-----------+---------------+-----------------+----------------+
|         3 | Ford | Fusion | 2023 |     85.00 | not available |               5 |              2 |
+-----------+------+--------+------+-----------+---------------+-----------------+----------------+
1 row in set (0.00 sec)
```

- **Let's add 5 customers in our database**



```
Enter your choice: 6
Enter first name: Ananya
Enter last name: Singh
Enter email: singh_ananya@gmail.com
Enter phone number: 4589763579
Customer added successfully.
```

```
Enter your choice: 6
Enter first name: Sanjay
Enter last name: Gupta
Enter email: gupta_sanjay@gmail.com
Enter phone number: 7891648536
Customer added successfully.
```

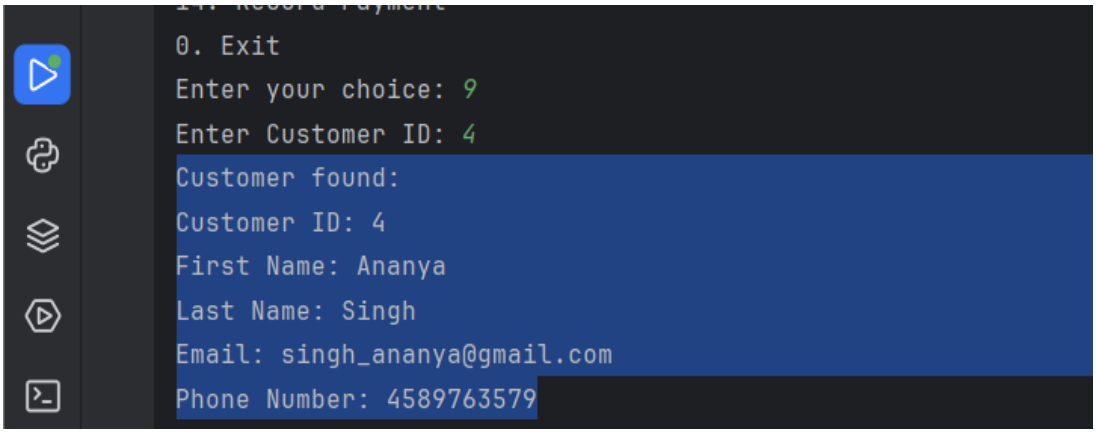
- **List all customers**

```
12. List Active Leases
13. List Lease History
14. Record Payment
0. Exit
Enter your choice: 8
Customers:
(1, 'Rajesh', 'Kumar', 'kumar_rajesh@gmail.com', '7865123489')
(2, 'Priya', 'Sharma', 'sharma_priya@gmail.com', '3256789468')
(3, 'Amit', 'Patel', 'amit.patel@gmail.com', '6548793897')
(4, 'Ananya', 'Singh', 'singh_ananya@gmail.com', '4589763579')
(5, 'Sanjay', 'Gupta', 'gupta_sanjay@gmail.com', '7891648536')
```

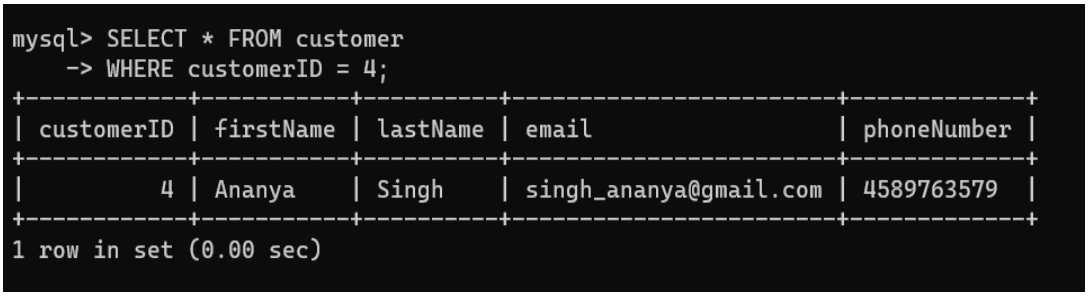
- **Let's check in MySQL Workbench**

```
mysql> SELECT * FROM customer;
+------------+-----------+----------+-----------------------+-------------+
| customerID | firstName | lastName | email                 | phoneNumber |
+------------+-----------+----------+-----------------------+-------------+
|          1 | Rajesh    | Kumar    | kumar_rajesh@gmail.com | 7865123489  |
|          2 | Priya     | Sharma   | sharma_priya@gmail.com | 3256789468  |
|          3 | Amit      | Patel    | amit.patel@gmail.com   | 6548793897  |
|          4 | Ananya    | Singh    | singh_ananya@gmail.com | 4589763579  |
|          5 | Sanjay    | Gupta    | gupta_sanjay@gmail.com | 7891648536  |
+------------+-----------+----------+-----------------------+-------------+
5 rows in set (0.01 sec)
```

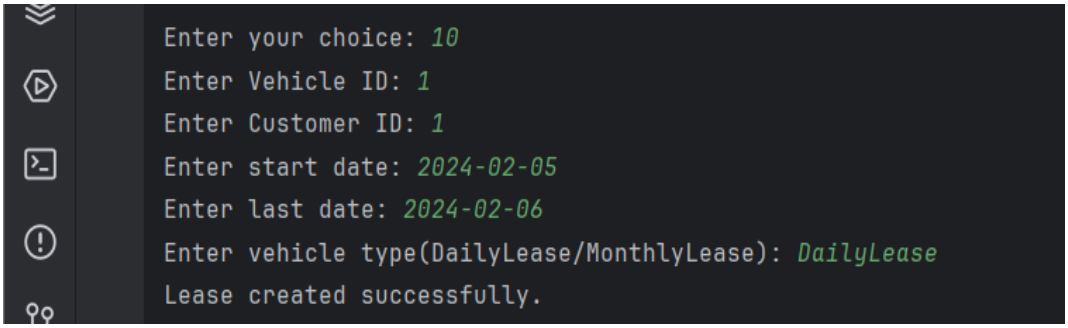
- **Let's select customer by customer ID**

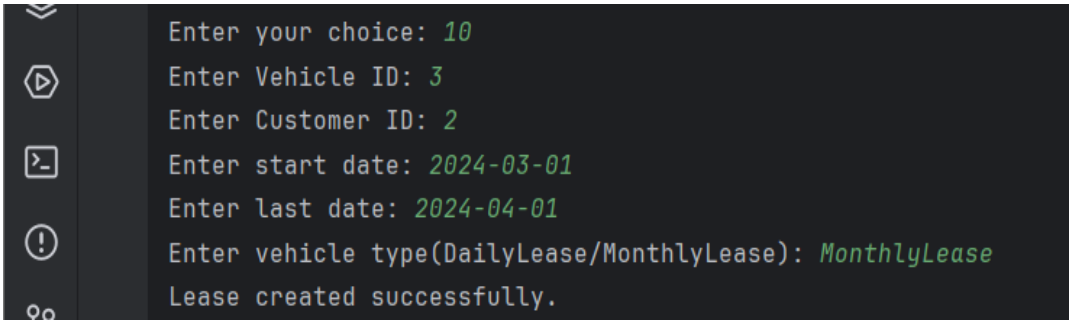


- **Let's check in MySQL Workbench**



- **Let's add 5 data in Lease Table**





```
Enter your choice: 10
Enter Vehicle ID: 4
Enter Customer ID: 3
Enter start date: 2024-02-07
Enter last date: 2024-02-08
Enter vehicle type(DailyLease/MonthlyLease): DailyLease
Lease created successfully.
```

```
Enter your choice: 10
Enter Vehicle ID: 5
Enter Customer ID: 4
Enter start date: 2024-03-27
Enter last date: 2024-04-27
Enter vehicle type(DailyLease/MonthlyLease): MonthlyLease
Lease created successfully.
```

```
Enter your choice: 10
Enter Vehicle ID: 2
Enter Customer ID: 5
Enter start date: 2024-02-26
Enter last date: 2024-02-27
Enter vehicle type(DailyLease/MonthlyLease): DailyLease
Lease created successfully.
```

- **List the Lease History**

```
Enter your choice: 13
Lease History:
Lease ID: 1, Vehicle ID: 1, Customer ID: 1, Start Date: 2024-02-05, End Date: 2024-02-06, Type: DailyLease
Lease ID: 2, Vehicle ID: 3, Customer ID: 2, Start Date: 2024-03-01, End Date: 2024-04-01, Type: MonthlyLease
Lease ID: 3, Vehicle ID: 4, Customer ID: 3, Start Date: 2024-02-07, End Date: 2024-02-08, Type: DailyLease
Lease ID: 4, Vehicle ID: 5, Customer ID: 4, Start Date: 2024-03-27, End Date: 2024-04-27, Type: MonthlyLease
Lease ID: 5, Vehicle ID: 2, Customer ID: 5, Start Date: 2024-02-26, End Date: 2024-02-27, Type: DailyLease
```

- **Let's check in MySQL Workbench**

```
mysql> select * from lease;
+---------+-----------+------------+------------+------------+--------------+
| leaseID | vehicleID | customerID | startDate  | endDate    | type         |
+---------+-----------+------------+------------+------------+--------------+
|       1 |         1 |          1 | 2024-02-05 | 2024-02-06 | DailyLease   |
|       2 |         3 |          2 | 2024-03-01 | 2024-04-01 | MonthlyLease |
|       3 |         4 |          3 | 2024-02-07 | 2024-02-08 | DailyLease   |
|       4 |         5 |          4 | 2024-03-27 | 2024-04-27 | MonthlyLease |
|       5 |         2 |          5 | 2024-02-26 | 2024-02-27 | DailyLease   |
+---------+-----------+------------+------------+------------+--------------+
5 rows in set (0.01 sec)
```

- **Active Lease**

```
Enter your choice: 12
Active Leases:

Lease ID: 3
Vehicle ID: 4
Customer ID: 3
Start Date: 2024-02-07
End Date: 2024-02-08
Type: DailyLease
```

- **Let's check in MySQL Workbench**

```
mysql> select * from lease;
+---------+-----------+------------+------------+------------+--------------+
| leaseID | vehicleID | customerID | startDate  | endDate    | type         |
+---------+-----------+------------+------------+------------+--------------+
|       1 |         1 |          1 | 2024-02-05 | 2024-02-06 | DailyLease   |
|       2 |         3 |          2 | 2024-03-01 | 2024-04-01 | MonthlyLease |
|       3 |         4 |          3 | 2024-02-07 | 2024-02-08 | DailyLease   |
|       4 |         5 |          4 | 2024-03-27 | 2024-04-27 | MonthlyLease |
|       5 |         2 |          5 | 2024-02-26 | 2024-02-27 | DailyLease   |
+---------+-----------+------------+------------+------------+--------------+
5 rows in set (0.01 sec)
```

- **Let's record 5 payments**

```
Enter your choice: 14
Enter Lease ID to record payment: 1
Enter date of payment: 2024-02-06
Enter payment amount: 175
Payment recorded successfully.
```

```
Enter your choice: 14
Enter Lease ID to record payment: 2
Enter date of payment: 2024-04-01
Enter payment amount: 55
Payment recorded successfully.
```

```
Enter your choice: 14
Enter Lease ID to record payment: 3
Enter date of payment: 2024-02-08
Enter payment amount: 85
Payment recorded successfully.
```

```
Enter your choice: 14
Enter Lease ID to record payment: 4
Enter date of payment: 2024-02-15
Enter payment amount: 45
Payment recorded successfully.
```

```
Enter your choice: 14
Enter Lease ID to record payment: 5
Enter date of payment: 2024-02-27
Enter payment amount: 58
Payment recorded successfully.
```

- **Let's check in MySQL Workbench**

```
mysql> select * from payment;
+-----------+---------+-------------+--------+
| paymentID | leaseID | paymentDate | amount |
+-----------+---------+-------------+--------+
|         1 |       1 | 2024-02-06  | 175.00 |
|         2 |       2 | 2024-04-01  |  55.00 |
|         3 |       3 | 2024-02-08  |  85.00 |
|         4 |       4 | 2024-02-15  |  45.00 |
|         5 |       5 | 2024-02-27  |  58.00 |
+-----------+---------+-------------+--------+
5 rows in set (0.01 sec)
```

- **Calculate lease cost**

```
15. Calculate lease cost
16. Get payment history for Customer
17. Calculate total revenue
18. Update customer
19. Record Payment
20. Find Lease by ID
0. Exit
Enter your choice: 15
Enter lease type (daily/monthly): monthly
Enter lease duration: 15
Enter daily rate: 75
Total lease cost: 33750.0
```

- **Payment history of a customer**

```
Enter your choice: 16
Enter Customer ID: 5
Payment History for Customer ID 5
payment_id: 5, lease_id: 5, payment_date: 2024-02-27, amount: 58.00
***** Welcome To Car Rental System *****
1. Add Car
```

- **Payment history of a customer**

```
17. Calculate total revenue
18. Update customer
19. Record Payment
20. Find Lease by ID
0. Exit
Enter your choice: 17
Total Revenue: 418.00
```

- **Find lease by ID**

```
Enter your choice: 19
Enter the lease ID: 5
lease ID: 5 vehicle ID 2 customer_id 5 start_date 2024-02-26 end_date 2024-02-27 vehicle_type DailyLease
***** Welcome To Car Rental System *****
```

- **Let's Update customer Info**

```
Enter your choice: 18
Enter Customer ID to update: 5
Enter new first name: new
Enter new last name: name
Enter new email: new_name@gmail.com
Enter new phone number: 8796541236
Customer information updated successfully.
***** Welcome To Car Rental System *****
```

- **Let's check in MySQL Workbench**

```
mysql> SELECT * FROM CUSTOMER WHERE customerID=5;
+------------+-----------+----------+--------------------+-------------+
| customerID | firstName | lastName | email              | phoneNumber |
+------------+-----------+----------+--------------------+-------------+
|          5 | new       | name     | new_name@gmail.com | 8796541236  |
+------------+-----------+----------+--------------------+-------------+
1 row in set (0.01 sec)
```

- **Let's Delete a customer record**

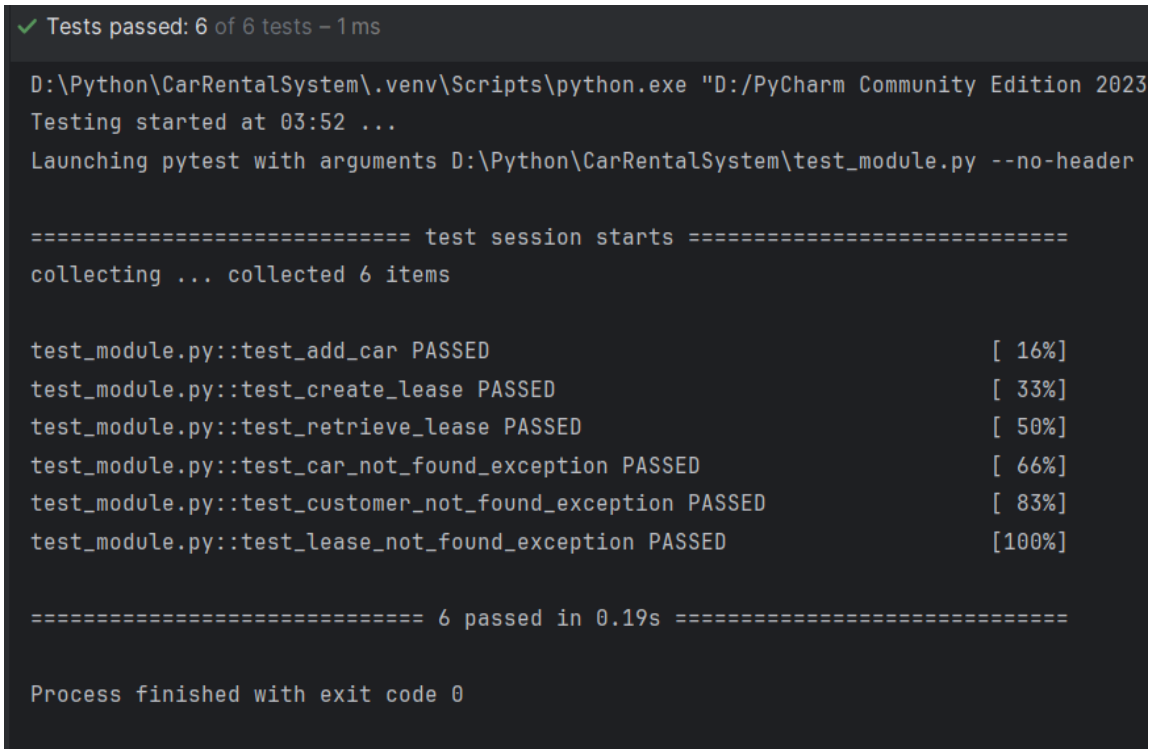
```
Enter your choice: 7
Enter customer ID to remove: 5
Customer removed successfully.
***** Welcome To Car Rental System *****
```

- **Corresponding record is deleted from the database**

Result Grid | Filter Rows: | Edit: | Export/Import:

customerID	firstName	lastName	email	phoneNumber
1	Rajesh	Kumar	kumar_rajesh@gmail.com	7865123489
2	Priya	Sharma	sharma_priya@gmail.com	3256789468
3	Amit	Patel	amit.patel@gmail.com	6548793897
4	Ananya	Singh	singh_ananya@gmail.com	4589763579
NULL	NULL	NULL	NULL	NULL

➢ Testing



---THE END---